A Simulation Data Driven Design Approach for Rapid Product Optimization

Yanli Shao[1], Huawei Zhu[2], Rui Wang[3], Ying Liu[4], Yusheng Liu[5*]

[1]Key Laboratory of Complex Systems Modeling and Simulation, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, PR.China

[2]Zhejiang University City College, Hangzhou 310015, PR.China

[3]Part Rolling Key Laboratory of Zhejiang Province, Ningbo University, Ningbo Zhejiang 315211, PR.China

[4]School of Engineering, Cardiff University, UK

[5]State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, PR.China

Abstract: Traditional design optimization is an iterative process of design, simulation and redesign, which requires extensive calculations and analysis. The designer needs to adjust and evaluate the design parameters manually and continually based on the simulation results until a satisfactory design is obtained. However, the expensive computational costs and large resource consumption of complex products hinder the wide application of simulation in industry. It is not an easy task to search the optimal design solution intelligently and efficiently. Therefore, a simulation data driven design approach which combines dynamic simulation data mining and design optimization is proposed to achieve this purpose in this study. The dynamic simulation data mining algorithm—Online Sequential Extreme Learning Machine with Adaptive Weights ($\mathrm{W_{adaptive}OS-ELM}$) is adopted to train the dynamic prediction model to effectively evaluate the merits of new design solutions in the optimization process. Meanwhile, the prediction model is updated incrementally by combining new 'good' dataset to reduce the modeling cost and improve the prediction accuracy. Furthermore, the improved heuristic optimization algorithm—Adaptive and Weighted Center Particle Swarm Optimization (AWCPSO) is introduced to guide the design change direction intelligently to improve the search efficiency. In this way, the optimal design solution can be searched automatically with less actual simulation iterations and higher optimization efficiency, and thus supporting the rapid product optimization effectively. The experimental results demonstrate the feasibility and effectiveness of the proposed approach.

Key words: Simulation Data driven design; Dynamic simulation data mining; Design optimization; $\mathrm{W_{adaptive}OS-ELM}$; Incremental learning; AWCPSO

1. Introduction

Traditional design optimization, which identifies the best combinations of design parameters under the given constraints to obtain more efficient and lighter structures, is of great significance for reducing the development cost and improving the performance of complex product [1,2,3]. Generally, the design parameters are adjusted continually and manually based on the simulation results until a satisfactory design solution is obtained during the loop of design-simulation-redesign. Since the simulation is computationally intensive, metamodel-based optimization attracted the attention of many researchers [2,3,4,5,6,7]. It is solved through popular optimization algorithms without actual simulations directly, which will reduce the computational cost and improve the optimization efficiency.

However, it is still difficult to search the optimal design solution efficiently and intelligently. The reasons may come from three folds. Firstly, the simulation, which requires a lot of expertise, experience and human intervention, is often extremely cumbersome and computationally intensive. Despite of the steady growth in computation power, the simulation complexity and cost is also growing fast and it is error-prone when the number of design variables increases. Especially, the design process is sensitive to the ability and skills of designers and analysts, thus making the design very unstable. Secondly, it is impossible for analysts to consider all possible scenarios or different variations even with required hardware resources since the design parameters are often numerous and changed continuously. Only a part of feasible parameter combinations can be simulated under certain conditions. It is necessary to introduce the optimization strategy to guide the search direction automatically and intelligently to improve the search efficiency and reduce the computational cost. Thirdly, many researchers have introduced the metamodel to approximate the relationship between the input design parameters and the desired performance parameters within acceptable accuracy. However, the approximation error between the metamodel and the real model cannot be avoided, thereby reducing its availability and reliability. Meanwhile, the metamodel generally does not consider the dynamic growth of the simulation data. It is necessary to adopt a more accurate model to simulate the complexity and dynamics of the simulation system. Accordingly, simulation and optimization should be considered and complementary to each other to facilitate the intelligent and efficient search of optimal design solution.

Based on the above analysis, a simulation data driven design approach, which combines the dynamic simulation data mining and design optimization based on the integration framework of Computer Aided Design (CAD) and Computer Aided Engineering (CAE), is proposed in this study. After the introduction of

the related work, the method overview is given in Section 3. And then, the dynamic simulation data mining algorithm—Online Sequential Extreme Learning Machine with Adaptive Weights ($\mathrm{W_{adaptive}OS-ELM}$) and the improved optimization algorithm—Adaptive and Weighted Center Particle Swarm Optimization (AWCPSO) are described in Section 4 and Section 5, respectively. Section 6 presents the related performance analysis and approach comparison. The conclusions and future work are finally offered in Section 7.

2. Related Work

Product design optimization has been a very challenging research issue in the field of engineering design, which requires not only a lot of experience and knowledge, but also some appropriate scientific approaches. In the traditional design optimization approach, designers usually adjust the design parameters manually based on the analysis results to assist the product optimization. However, it is relatively complicated, time-consuming and cannot be executed automatically. This leads to the design quality that depends on individual capabilities, which in turn reduces the robustness of the final design. In the past decades, simulation-based optimization methods [2,3,8] have been extensively studied, especially the optimization techniques in the CAD/CAE integration framework. The optimization techniques mainly include gradient-based method, evolutionary algorithm, approximate optimization algorithm, etc. Every optimization technique has its own characteristics and scope of application.

Hare and Dai et al. [9,10] applied the gradient-based method to provide the direct coupling between CAE software and the optimization algorithm to search the optimal solution. However, the gradient information is computationally expensive or even unavailable in many cases, and it is sometimes easy to fall into local optimum. In addition, many evolutionary algorithms (EAs) such as genetic algorithm (GA) [11,12], particle swarm optimization (PSO) algorithm [13,14,15], immune algorithm [16] and artificial bee colony algorithm (ABC) [17] are coupled directly with CAE software for structural optimization design as well. Compared with the gradient-based algorithm, EAs, which is independent of gradient information, has greater stability, applicability and global optimization ability. However, the above methods require a lot of performance evaluations performed by computation-intensive simulation, especially for complex products, which leads to lower iterative optimization efficiency and higher cost [11].

Recently, approximation techniques such as response surface method (RSM) [2,3,5,6], radial basis function (RBF) [2] and sequential approximate optimization (SAO) [3,18] are extensively explored to address

the structural optimization design issues. In these algorithms, approximation technique is used to construct the approximation model or metamodel, and then the metamodel is combined with GA or PSO to get the optimal solution. Wherein, the CAD/CAE integration framework [2,3] is proposed to enable the design optimization process to be seamlessly cycled and automatically without manual intervention to free the designer from the repetitive tasks, and thus improving the design efficiency and the product quality. Park and Dang [2] applied metamodeling techniques including RSM and RBF to the structural optimization. RSM is suitable and effective in engineering design applications when the number of design parameters is relatively small and the response is not highly linear, otherwise RBF is more suitable. Wang et al. [3] presents an integrated framework that performs the structural design optimization by associating the improved SAO algorithm with the CAD/CAE integration technique. The SAO algorithm is especially suitable for computation-intensive structural design issues since less function evaluations are needed in comparison with pure EAs, which greatly reduce the computational cost and improve the optimization efficiency. However, as mentioned previously, more accurate models are needed to simulate the complexities and dynamics of the simulation system since the error between the approximation model and real model cannot be avoided [19].

Furthermore, data mining (DM) based approach [20], which uses data mining technique as an effective tool, has been applied to product design optimization recently. Chen and Huang [20] proposed a hybrid global optimization algorithm that combines DM, evolutionary strategy (ES) and sequential quadratic programming (SQP) to search for the global optimal solution effectively. The search space is reduced through the classification, association or clustering activities in the DM algorithm to increase the possibility of finding the global optimal solution. Better et al. [21] proposed a more comprehensive and integrated model framework based on the combined dynamic data mining and simulation optimization techniques. It is used on a market research application to illustrate its feasibility and effectiveness, but it is limited to social science case studies. After that, Li and Roy [22] applied this idea to the manufacturing area. They brings three distinct system paradigms—data analytics, simulation and optimization, into one to provide a computational platform. However, due to the high complexity and uncertainty of the real-world system, how to effectively analyze and execute the simulation process to improve the system performance is still an open problem.

Based on the above analysis, three key issues that need to be solved in product design optimization are: (1) how to automate the iterative process of design-simulation-optimization-redesign; (2) how to dynamically simulate the real world with high complexity and uncertainty as accurately as possible; (3) how to reduce the computational cost and improve the iterative optimization efficiency under the premise of efficient and

accurate performance evaluation of new design scheme. The first question solved through the CAD/CAE integration framework by using common scripting, programming languages and Application Programming Interface (API) has been studied by some researchers. The last two issues require further study to realize the intelligent and efficient search of optimal solution. This research is motivated by this gap and aims to develop a methodology to tackle the aforementioned challenges to support the rapid product optimization.

3. Method Overview

To cope with all the above issues, a simulation data driven design approach which combines dynamic simulation data mining and design optimization based on the CAD/CAE integration framework is proposed. The main idea is: (1) the optimization strategy is introduced to guide the design change direction automatically to improve the search efficiency by reducing the design search space, which is suitable for solving global optimization problems with implicit relationship; (2) the dynamic prediction model is adopted to evaluate the performance of new design schemes in the process of optimization evaluation. Only better or optimal design schemes are simulated to fully verify the product performance, which will greatly reduce the number of actual simulations and improve the computational efficiency; (3) the prediction model is updated incrementally by combining new 'good' simulation dataset obtained from optimization, on the basis of reusing the original prediction model to reduce the modeling cost and improve the prediction accuracy. It is of great significance for computation-intensive structural design optimization problem since fewer actual function evaluations are required to seek the optimum. In this way, the optimal design solution can be searched automatically with less actual simulation iterations and higher optimization efficiency.

After the selection of dynamic simulation data mining and design optimization technique, the proposed approach which combines these two will be executed automatically without manual intervention based on the customized CAD/CAE integration framework. The method overview is given in Fig. 1, and the specific details outlined below.

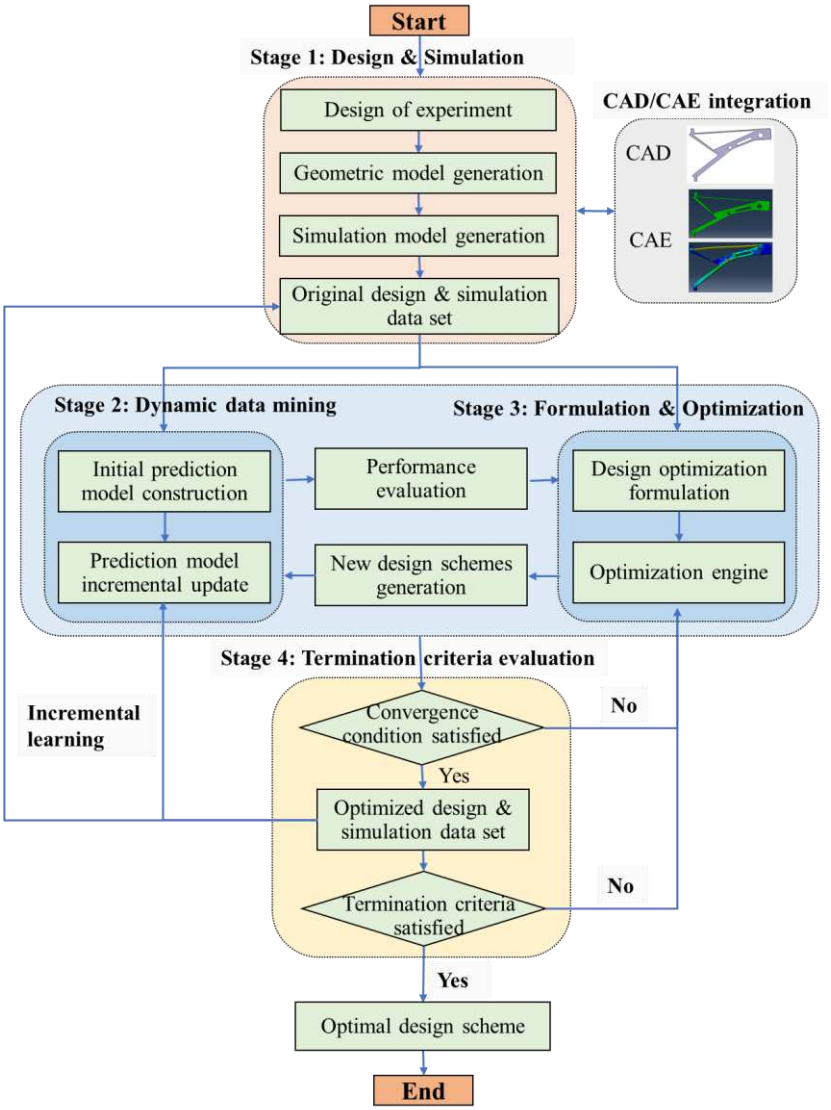


Fig. 1 Method overview.

Stage 1: Design and Simulation. The geometric models were designed in the CAD software based on the design of experiment (DOE), such as comprehensive experiment or orthogonal experiment. The simulations were performed in the CAE software to obtain the corresponding simulation data. The whole process is fully conducted in the CAD/CAE integration framework automatically.

Stage 2: Dynamic data mining. Original design and simulation data set are provided to construct the prediction model for the subsequent performance evaluation. It is used to measure the merits of new design schemes when combined with optimization module to decrease actual simulation. Moreover, the prediction model is dynamically updated through incremental learning algorithm by combining new 'good' dataset obtained from the optimization engine.

Stage 3: Formulation and Optimization. For the product design optimization problem, the objective function, decision variables, and constraints should be identified firstly. Then it is converted to an optimal

search problem under the constraint condition after formulation. Then the training data set is used as the initial samples to submit to the optimization engine. The optimization algorithm is applied to guide the right design change direction to generate new 'good' design schemes to reduce search space.

Stage 4: Termination criteria evaluation. The iterative optimization process continues until the predetermined convergence condition (i.e., the maximum number of iterations) is satisfied. Some better design schemes are simulated to get additional data set to dynamically update the prediction model through incremental learning algorithm. If the prediction error between the predicted and real simulation results of optimal design solution is less than a given threshold, while the real simulation results satisfy a predetermined optimization goal, the iterative process stops and outputs the optimal design scheme. Otherwise, dynamic data mining and optimization steps are repeated until the satisfactory design solution is obtained.

Here, the dynamic prediction model and the optimization algorithm should be chosen carefully as they affect the optimization accuracy and efficiency. Their details are described in the following two sections.

4. Incremental Learning based Dynamic Data Mining Algorithm-$W_{adaptive}$OS-ELM

4.1. Problem analysis

Generally, varieties of simulation systems are involved in the simulation tasks for complex products to satisfy the needs of domain-specific simulations, which leads to a growing number of simulation data and brings about the challenges of massive simulation data management [23]. Therefore, simulation data mining techniques are applied to discover the knowledge and rules implied in the large simulation data for performance evaluation. However, the simulation data are often dynamically updated during the iterative design optimization process and thus static simulation data mining cannot capture the dynamic nature of a simulation system in real time. Especially, batch learning is limited in most cases since the complete data set is usually not available at once. It is also time-consuming and computationally expensive to reconstruct the prediction model from scratch as it repeats the training with the past data as well as the newly arrived data without any difference. Especially, as the complexity of the problem and the size of the sample data set grow, the space-time demand will increase rapidly, which will eventually lead to longer training time and poorer prediction effects. How to reuse the dynamic simulation data to simulate the dynamics and complexities of the simulation system quickly and accurately is of great significance. Therefore, dynamic simulation data mining technique is proposed and applied to the dynamically updated simulation data. On the basis of original

prediction model, the dynamic prediction model here should support incremental learning, so as to reduce the modeling complexity and the overhead of repetitive learning as well as improve the prediction efficiency and accuracy.

Incremental learning [24] means continual learning of new knowledge from newly arrived samples, which enables the system to be self-adaptive to the newly added data without retraining the learning model from scratch, thereby reducing spatiotemporal requirements to better satisfy the actual needs. The frequently used incremental learning algorithms include Support Vector Machine (SVM) [24,25], Recursive Least Squares (RLS) [26], On-line Sequential Extreme Learning Machine (OS-ELM) [27] and ensemble learning algorithm such as On-line Weighted Ensemble (OWE) [28], Dynamic and On-line Ensemble Regression (DOER) [29], Online Accuracy Updated Ensemble (OAUE) [30], etc.

Among them, OS-ELM, as a fast and accurate incremental learning algorithm for non-linear problem on the basis of Extreme Learning Machine (ELM) [31], has many excellent characteristics, such as faster learning speed, good generalization performance and relatively simple realization process when compared to other popular sequential learning algorithms. Currently, a lot of research work on OS-ELM improvement from various aspects has been proposed. Some are improvements to OS-ELM itself [32,33,34], and others are research on the ensemble algorithm of OS-ELM [35]. The proposed algorithm in this study belong to the former. All of them have their own characteristics and scope of applications, and their performance needs to be verified in specific problem and a large number of experiments.

Similar to ELM, the input weights and biases are randomly assigned without tuning during the learning process of OS-ELM, and the Least Squares (LS) method is used to determine the output weight vector $\boldsymbol{\beta}$ as well. However, the estimation error cannot be ensured to be the best even when the minimum estimation mean square error of the measurement is the smallest. The measurement accuracy may not be high as the measurement values are used regardless of their merits. Therefore, more accurate prediction model would be constructed if different samples are weighted based on their respective quality. Based on the above analysis, an improved OS-ELM with adaptive weights ($\mathrm{W_{adaptive}}$OS-ELM) is proposed to construct a more accurate dynamic prediction model in this study.

4.2. The $\mathbf{W_{adaptive}}$OS-ELM algorithm

OS-ELM, developed by Liang et al. [27] is reviewed firstly to provide the necessary background. Then

the algorithm principle and learning algorithm for $W_{adaptive}$OS-ELM are given in details.

4.2.1. Brief review of OS-ELM

As mentioned above, OS-ELM originates from the batch learning ELM (as shown in Fig. 2), is an on-line algorithm for single layer feedforward network(SLFN) that can learn data on a sample or batch basis.

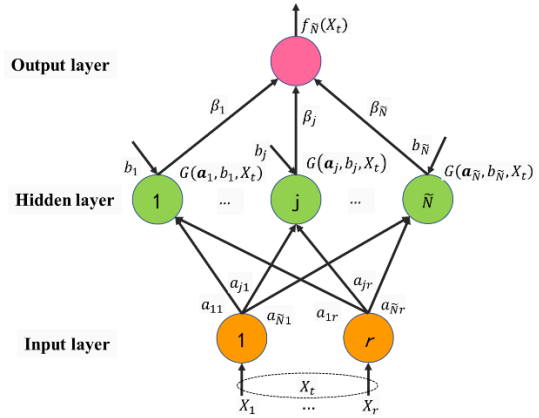


Fig. 2. ELM structure.

For a data set $\mathbf{D} = \{(\boldsymbol{X}_t, y_t)\}_{t=1}^{N}$ with N distinct samples, the output of j-th hidden node with respect to the input $\mathbf{X}$ is activation function $G(a_j, b_j, \boldsymbol{X}_t)$, and its corresponding parameters of hidden nodes are $\{a_j, b_j\}_{j=1}^{\tilde{N}}$ with $\tilde{N}$ hidden nodes (additive (Sigmoid) or radial basis function (RBF) nodes). $\boldsymbol{\beta}_j$ connects the j-th hidden node and the output node. The hidden layer is to provide a complex network structure to deal with the problem that low-dimensional structure cannot handle. If an SLFN with $\tilde{N}$ hidden nodes can approximate N samples of $\mathbf{D}$ with zero error, there exist a_j, b_j and $\boldsymbol{\beta}_j$ such that:

$$f_{\tilde{N}}(\boldsymbol{X}_t) = \sum_{j=1}^{\tilde{N}} \beta_j G(\boldsymbol{a}_j, b_j, \boldsymbol{X}_t) = y_t (t = 1, \dots, N) \tag{1}$$

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{y} \tag{2}$$

$$\mathbf{H} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_1) & \cdots & G(\boldsymbol{a}_{\tilde{N}}, b_{\tilde{N}}, \boldsymbol{X}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_N) & \cdots & G(\boldsymbol{a}_{\tilde{N}}, b_{\tilde{N}}, \boldsymbol{X}_N) \end{bmatrix}_{N \times \tilde{N}} \tag{3}$$

Here, $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{\tilde{N}}]^{\mathrm{T}}, \mathbf{y} = [y_1, \dots, y_{\tilde{N}}]^{\mathrm{T}}, \boldsymbol{a}_j = \left[a_{j1}, a_{j2}, \dots, a_{jr}\right]^{T}, \boldsymbol{b} = [b_1, \dots, b_{\tilde{N}}]^{\mathrm{T}}$, $\boldsymbol{\beta}$ is the output weight vector and $\mathbf{y}$ is the output vector. $\mathbf{H}$ is the hidden layer output matrix. Where, the j-th column represents the j-th hidden node output vector with respect to all the inputs, and the t-th row represents the output vector of the hidden layer with respect to $\boldsymbol{X}_t$.

The OS-ELM algorithm consists of two phases. In the initialization phase, the initial training data set $\mathbf{D}_0 = \{(\boldsymbol{X}_t, y_t)\}_{t=0}^{N_0} \left(\widetilde{N} \le N_0 < N\right)$ is selected from $\mathbf{D}$ to train the initial prediction model $\boldsymbol{f_0}$. In the sequential learning phase, on-line samples are employed either one-by-one or chunk-by-chunk (with fixed or varying size) for on-line retraining, where the (*k+1*)–th ($k \ge 0$) chunk data set $\mathbf{D}_{k+1}$ with N_{k+1} samples is defined as follows:

$$\mathbf{D}_{k+1} = \{(\boldsymbol{X}_t, y_t)\}_{t=\sum_{l=0}^{k} N_l + 1}^{t=\sum_{l=0}^{k+1} N_l} (\mathrm{t} = N_0 + 1, \ldots, N_0 + \cdots + N_{k+1}) \tag{4}$$

Generally, $\boldsymbol{\beta_0}$ can be estimated by using pseudo inverse of $\mathbf{H_0}$ with a small nonzero training error $\epsilon > 0$ as

$$\widehat{\boldsymbol{\beta_0}} = \mathbf{H_0}^{\dagger} \boldsymbol{y_0} \tag{5}$$

$$\mathbf{H_0}^{\dagger} = \left(\mathbf{H_0}^{\mathbf{T}} \mathbf{H_0}\right)^{-1} \mathbf{H_0}^{\mathbf{T}} \tag{6}$$

$$\mathbf{H_0} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_1) & \cdots & G(\boldsymbol{a}_{\widetilde{N}}, b_{\widetilde{N}}, \boldsymbol{X}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_{N_0}) & \cdots & G(\boldsymbol{a}_{\widetilde{N}}, b_{\widetilde{N}}, \boldsymbol{X}_{N_0}) \end{bmatrix}_{N_0 \times \widetilde{N}} \tag{7}$$

Here, $\mathbf{H_0}^{\dagger}$ is the Moore-Penrose generalized inverse [36] of $\mathbf{H_0}$. If $\mathbf{H_0}^{\mathbf{T}}\mathbf{H_0}$ is nonsingular and its inverse exists, $\mathbf{H_0}^{\dagger}$ is calculated based on Eq. (6), where $\boldsymbol{P_0} = \left(\mathbf{H_0}^{\mathbf{T}} \mathbf{H_0}\right)^{-1}$ and $\boldsymbol{\beta_0}$ is estimated as:

$$\boldsymbol{\beta_0} = \left(\mathbf{H_0}^{\mathbf{T}} \mathbf{H_0}\right)^{-1} \mathbf{H_0}^{\mathbf{T}} \boldsymbol{y_0} \tag{8}$$

After that, with the arrival of *(k+1)*-th chunk with $\boldsymbol{N_{k+1}}$ samples, $\mathbf{H_{k+1}}$, $\boldsymbol{\beta_{k+1}}$ and $\boldsymbol{y_{k+1}}$ are computed as follows:

$$\mathbf{H_{k+1}} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_{(\sum_{l=0}^{k} N_l)+1}) & \cdots & G(\boldsymbol{a}_{\widetilde{N}}, b_{\widetilde{N}}, \boldsymbol{X}_{(\sum_{l=0}^{k} N_l)+1})) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{X}_{\sum_{l=0}^{k+1} N_l}) & \cdots & G(\boldsymbol{a}_{\widetilde{N}}, b_{\widetilde{N}}, \boldsymbol{X}_{\sum_{l=0}^{k+1} N_l}) \end{bmatrix}_{N_{k+1} \times \widetilde{N}} \tag{9}$$

$$\boldsymbol{\beta_{k+1}} = \boldsymbol{\beta_k} + \boldsymbol{P_{k+1}} \mathbf{H_{k+1}}^{\mathbf{T}} (\boldsymbol{y_{k+1}} - \mathbf{H_{k+1}} \boldsymbol{\beta_k}) \tag{10}$$

$$\boldsymbol{P_{k+1}} = \boldsymbol{P_k} - \boldsymbol{P_k} \mathbf{H_{k+1}}^{\mathbf{T}} \left(\mathbf{I} + \mathbf{H_{k+1}} \boldsymbol{P_k} \mathbf{H_{k+1}}^{\mathbf{T}}\right)^{-1} \mathbf{H_{k+1}} \boldsymbol{P_k} \tag{11}$$

$$\boldsymbol{y_{k+1}} = \left[y_{(\sum_{l=0}^{k} N_l)+1}, \ldots, y_{\sum_{l=0}^{k+1} N_l}\right]^{\mathrm{T}} \tag{12}$$

4.2.2. The algorithm principle for $\boldsymbol{W_{adaptive}}$OS-ELM

The core idea of $\mathrm{W_{adaptive}}$OS-ELM is to adopt the instance weighting (IW) strategy in the on-line sequential learning phase, that is, each new sample is dynamically weighted according to its contribution to

the optimization goal and its prediction error on current prediction model. Here, the weighted least squares (WLS) [37] method is adopted to solve the output weight vector $\boldsymbol{\beta}$ to obtain a more accurate model for dynamic prediction. Based on the following **Theorem 1**, if $\mathbf{H^TWH}$ is invertible, $\boldsymbol{\beta}$ in $\mathrm{W_{adaptive}}$OS-ELM can be calculated as $\boldsymbol{\beta} = \left(\mathbf{H^TWH}\right)^{-1}\mathbf{H^TW}\boldsymbol{y}$.

Theorem 1: Let $\mathbf{H^TWH}$ be invertible, the WLS estimate of the parameter $\boldsymbol{x}$ based on the measurement information $\boldsymbol{z}$ and the weight matrix $\mathbf{W}$ is $\hat{\boldsymbol{x}}_{WLS} = \left(\mathbf{H^TWH}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{W}\boldsymbol{z}$.

Proof:

$$\min(\boldsymbol{z}-\mathbf{H}\boldsymbol{x})^{\mathbf{T}}\mathbf{W}(\boldsymbol{z}-\mathbf{H}\boldsymbol{x}) = \min\left(\boldsymbol{z}^{\mathbf{T}}\mathbf{W}\boldsymbol{z} - 2\boldsymbol{z}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x} + \boldsymbol{x}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x}\right)$$

$$\frac{\partial}{\partial \boldsymbol{x}}\left(\boldsymbol{z}^{\mathbf{T}}\mathbf{W}\boldsymbol{z} - 2\boldsymbol{z}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x} + \boldsymbol{x}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x}\right) = \frac{\partial}{\partial \boldsymbol{x}}\left(\boldsymbol{z}^{\mathbf{T}}\mathbf{W}\boldsymbol{z} - 2\boldsymbol{x}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}\mathbf{W}\boldsymbol{z} + \boldsymbol{x}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x}\right)$$

$$= -2\mathbf{H}^{\mathbf{T}}\mathbf{W}\boldsymbol{z} + 2\mathbf{H}^{\mathbf{T}}\mathbf{WH}\boldsymbol{x} = \mathbf{0}$$

$$\therefore \hat{\boldsymbol{x}}_{WLS} = \left(\mathbf{H^TWH}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{W}\boldsymbol{z}$$

(Note: $\frac{\partial <\mathbf{Ax},\mathbf{x}>}{\partial \mathbf{x}} = \boldsymbol{Ax} + \boldsymbol{A^T x}, \frac{\partial <\boldsymbol{x},\boldsymbol{y}>}{\partial \boldsymbol{x}} = \boldsymbol{y}$ **)**

As the samples with greater optimization contribution and smaller prediction error have higher quality, they should be given more attention with larger weight to build a more accurate prediction model in the training process. Here, there is a hypothesis that the smaller the optimization goal value is, the better the sample is. Let m denotes the sample size of training data set, the weight d_t of each sample X_t is computed as follows:

(1) Contribution weight d_{ot} computation. Generally, the smaller the optimization goal value $F(X_t)$, the larger the weight d_{ot}.

$$d_{ot} = \left(\sum_{t=1}^{m} F(X_t) - F(X_t)\right) \Big/ \sum_{t=1}^{m} F(X_t)(t = 1, \ldots, m) \tag{13}$$

For example, when the optimization goal is to minimize the model volume to reduce the material consumption. $F(X_t)$ represents the model volume value, and then $\sum_{t=1}^{m} F(X_t)$ is the sum value of m samples. Correspondingly, sample X_t with smaller volume value should be paid more attention with larger weight d_{ot} among all the samples to build more accurate prediction model.

(2) Prediction error weight d_{et} computation. The relative prediction error computed by the prediction model $\boldsymbol{f_0}$ is $\hat{e}_t = |\,(y_t - \mathbf{H_k}\boldsymbol{\beta_k})/y_t\,|$. Then d_{et} is defined as:

$$d_{et} = \left(\sum_{t=1}^{m} \hat{e}_t - \hat{e}_t\right) \Big/ \sum_{t=1}^{m} \hat{e}_t\ (t = 1, \ldots, m) \tag{14}$$

Therefore, the sample weight d_t $(0 \le \gamma,\ \delta \le 1,\ \gamma + \delta = 1)$ is computed as $d_t = \gamma \times d_{ot} + \delta \times d_{et}$. Where γ and δ represent the weight of d_{ot} and d_{et}, respectively. Their settings depend on the user's needs, and (γ, δ) is set to be (0.5,0.5) in this study.

4.2.3. The learning algorithm for $\boldsymbol{W_{adaptive}}$OS-ELM

Similar to OS-ELM, $W_{adaptive}$OS-ELM is implemented in two phases. In the initialization phase, the prediction model is built based on the initial training data set, and it is updated dynamically in the on-line sequential based on the newly added samples. According to the standard ELM algorithm and the WLS method, the output weight vector $\boldsymbol{\beta_0}$ in the initialization phase is calculated as:

$$\boldsymbol{\beta_0} = \left(\mathbf{H_0}^{\mathbf{T}} \boldsymbol{W_0} \mathbf{H_0}\right)^{-1} \mathbf{H_0}^{\mathbf{T}} \boldsymbol{W_0} \boldsymbol{y_0} \tag{15}$$

$\mathbf{H_0}$ and $\mathbf{W_0}$ are the initial hidden layer output matrix and the weight matrix, respectively. $\boldsymbol{y_0} = \left[y_1, \ldots, y_{N_0}\right]^{\mathrm{T}}$ is the corresponding initial output vector. If $\mathbf{W_0}(= \mathbf{I})$ is a symmetric positive definite weight matrix, that is, the standard ELM algorithm. With the arrival of *(k+1)*-th ($k = 0$) chunk with $\boldsymbol{N_1}$ samples, the new output weight vector $\boldsymbol{\beta_1}$ is:

$$\boldsymbol{\beta_1} = \left(\begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix}\boldsymbol{W_0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{W_1}\end{bmatrix} \begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix}\boldsymbol{W_0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{W_1}\end{bmatrix} \begin{bmatrix}\mathbf{y_0}\\ \mathbf{y_1}\end{bmatrix} \tag{16}$$

Here, $\mathbf{W_1} = \begin{bmatrix}\boldsymbol{d_{N_0+1}} & \cdots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \cdots & \boldsymbol{d_{N_0+N_1}}\end{bmatrix}$. Let $\boldsymbol{Q_0} = \mathbf{H_0}^{\mathbf{T}} \mathbf{W_0} \mathbf{H_0}, \boldsymbol{\beta_0} = \boldsymbol{Q_0}^{-1} \mathbf{H_0}^{\mathbf{T}} \mathbf{W_0} \boldsymbol{y_0},\ \boldsymbol{Q_1}$ is calculated as follows:

$$\boldsymbol{Q_1} = \begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix}\mathbf{W_0} & \mathbf{0}\\ \mathbf{0} & \mathbf{W_1}\end{bmatrix} \begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix} = \mathbf{H_0}^{\mathbf{T}} \mathbf{W_0} \mathbf{H_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{H_1} = \mathbf{Q_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{H_1} \tag{17}$$

According to Eq.(15) and Eq.(16-17), $\begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}^{\mathbf{T}} \begin{bmatrix}\mathbf{W_0} & \mathbf{0}\\ \mathbf{0} & \mathbf{W_1}\end{bmatrix} \begin{bmatrix}\mathbf{y_0}\\ \mathbf{y_1}\end{bmatrix}$ can be rewritten as

$$\begin{aligned} &\begin{bmatrix}\mathbf{H_0}\\ \mathbf{H_1}\end{bmatrix}^{\mathbf{T}} \begin{bmatrix}\mathbf{W_0} & \mathbf{0}\\ \mathbf{0} & \mathbf{W_1}\end{bmatrix} \begin{bmatrix}\mathbf{y_0}\\ \mathbf{y_1}\end{bmatrix} = \mathbf{H_0}^{\mathbf{T}} \mathbf{W_0} \mathbf{y_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{y_1}\\ &= \mathbf{Q_0} \boldsymbol{\beta_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{y_1} = \mathbf{Q_1} \boldsymbol{\beta_0} - \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{H_1} \boldsymbol{\beta_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{y_1} \end{aligned} \tag{18}$$

Based on Eq.(16) and Eq.(18), $\boldsymbol{\beta_1}$ is recomputed as:

$$\boldsymbol{\beta_1} = \mathbf{Q_1}^{-1}\left(\mathbf{Q_1} \boldsymbol{\beta_0} - \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{H_1} \boldsymbol{\beta_0} + \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} \mathbf{y_1}\right) = \boldsymbol{\beta_0} + \boldsymbol{Q_1}^{-1} \mathbf{H_1}^{\mathbf{T}} \mathbf{W_1} (\mathbf{y_1} - \mathbf{H_1} \boldsymbol{\beta_0}) \tag{19}$$

To generalize the above derivation, with the arrival of *(k+1)*-th ($k > 0$) chunk with $\boldsymbol{N_{k+1}}$ samples, $\boldsymbol{Q_{k+1}}^{-1}$ rather than $\boldsymbol{Q_{k+1}}$ is used to compute the new output weight vector $\boldsymbol{\beta_{k+1}}$ in order to avoid inverse

calculation in the recursive process. Here, Woodbury formula [38] is used to derive $\boldsymbol{Q_{k+1}}^{-1}$ as follows:

$$\begin{aligned}\boldsymbol{Q_{k+1}}^{-1} &= \left(\boldsymbol{Q_k} + \boldsymbol{H_{k+1}}^{T}\boldsymbol{W_{k+1}}\boldsymbol{H_{k+1}}\right)^{-1} \\ &= \boldsymbol{Q_k}^{-1} - \boldsymbol{Q_k}^{-1}\boldsymbol{H_{k+1}}^{T}\left(\boldsymbol{W_{k+1}}^{-1} + \boldsymbol{H_{k+1}}\boldsymbol{Q_k}^{-1}\boldsymbol{H_{k+1}}^{T}\right)^{-1}\boldsymbol{H_{k+1}}\boldsymbol{Q_k}^{-1}\end{aligned} \tag{20}$$

Let $\boldsymbol{P_{k+1}} = \boldsymbol{Q_{k+1}}^{-1}$, $\boldsymbol{\beta_{k+1}}$ can be rewritten as:

$$\boldsymbol{\beta_{k+1}} = \boldsymbol{\beta_k} + \boldsymbol{P_{k+1}}\mathbf{H_{k+1}}^{\mathrm{T}}\boldsymbol{W_{k+1}}(\boldsymbol{y_{k+1}} - \mathbf{H_{k+1}}\boldsymbol{\beta_k}) \tag{21}$$

$$\boldsymbol{P_{k+1}} = \boldsymbol{P_k} - \boldsymbol{P_k}\mathbf{H_{k+1}}^{\mathrm{T}}\left(\boldsymbol{W_{k+1}}^{-1} + \mathbf{H_{k+1}}\boldsymbol{P_k}\mathbf{H_{k+1}}^{\mathrm{T}}\right)^{-1}\mathbf{H_{k+1}}\boldsymbol{P_k} \tag{22}$$

In particularly, $\widetilde{N}$ different samples should be included in $\mathbf{D}_0$ to make rank($\boldsymbol{H_0}$)= $\widetilde{N}$ to ensure that $\mathbf{H_0}^{\mathrm{T}}\mathbf{H_0}$ is invertible. If $N_k \equiv 1$, it means that only one sample in the *(k+1)*-th chunk. Then Eq.(21-22) can be written as:

$$\boldsymbol{\beta_{k+1}} = \boldsymbol{\beta_k} + \boldsymbol{P_{k+1}}\mathbf{h_{k+1}}^{\mathrm{T}}\boldsymbol{w_{k+1}}(\boldsymbol{y_{k+1}} - \mathbf{h_{k+1}}\boldsymbol{\beta_k}) \tag{23}$$

$$\boldsymbol{P_{k+1}} = \boldsymbol{P_k} - \frac{\boldsymbol{P_k}\mathbf{h_{k+1}}^{\mathrm{T}}\mathbf{h_{k+1}}\boldsymbol{P_k}}{\boldsymbol{w_{k+1}}^{-1} + \mathbf{h_{k+1}}\boldsymbol{P_k}\mathbf{h_{k+1}}^{\mathrm{T}}} \tag{24}$$

$$\boldsymbol{h_{k+1}} = \left[G\left(\boldsymbol{a}_1, b_1, \boldsymbol{X}_{\left(\sum_{l=0}^{k} N_l\right)+1}\right), \ldots, G(\boldsymbol{a}_{\widetilde{N}}, b_{\widetilde{N}}, \boldsymbol{X}_{\left(\sum_{l=0}^{k} N_l\right)+1})\right] \tag{25}$$

Based on the above analysis, the learning algorithm for $W_{adaptive}$OS-ELM is shown in **Algorithm 1**. It is noteworthy that the sample weight matrix $\mathbf{W_0}$ can be recalculated based on the sample contributions to the optimization goal and prediction error after initial training to rebuild a more accurate initial prediction model.

Algorithm 1: Learning algorithm for $\mathrm{W_{adaptive}}$OS-ELM.

Input: Training data sets $\mathbf{D} = \{(\boldsymbol{X}_t, y_t)\}_{t=1}^{N}$ with N different samples; an activation function g(·); initial training data set $\mathbf{D}_0 = \{(\boldsymbol{X}_t, y_t) | \boldsymbol{X}_t \in R^{r\times 1}, y_t \in R\}_{t=1}^{N_0}$ with N_0 samples; the number of hidden nodes $\widetilde{N}(\widetilde{N} \le N_0 < N)$;

1. **Initialization phase**: initial prediction model training based on initial training data set $\mathbf{D}_0 = \{(\boldsymbol{X}_t, y_t) | \boldsymbol{X}_t \in R^{r\times 1}, y_t \in R\}_{t=1}^{N_0}$.
   (a) Initialize the weights of all the samples $d_t(t = 1, \ldots, N_0)$ as 1 in the $\mathbf{D}_0$, and set the initial weight matrix $\mathbf{W_0} = \mathbf{I}$.
   (b) Randomly assign input weight vector $\boldsymbol{a}_j$ and bias $\boldsymbol{b}_j\left(\mathrm{j} = 1, \ldots, \widetilde{N}\right)$;
   (c) Calculate matrix $\mathbf{H_0}$ using $\mathbf{D}_0$, g(x), $\boldsymbol{a}_j$, $\boldsymbol{b}_j$ based on Eq.(7);

(d) Calculate the output weight vector $\boldsymbol{\beta_0}$ through Eq.(15);

(e) Calculate the output $\boldsymbol{y_0} = \mathbf{H_0}\boldsymbol{\beta_0}$ to obtain initial model $\boldsymbol{f_0}$;

(f) Set $\mathrm{k} = 0, t = m = N_0$;

2. **On-line sequential learning phase**: prediction model incremental learning when new samples are added sequentially.

(a) Set $t = t + 1;\ m = m + 1$;

(b) Obtain a new sample $(\boldsymbol{X}_t, y_t)$ from $\mathbf{D}$ and set $\mathbf{D}_{k+1} = \mathbf{D}_k + (\boldsymbol{X}_t, y_t)$, calculate sample weight $\boldsymbol{w_{k+1}} = d_t$ based on Eq.(13-14);

(c) Obtain matrix $\boldsymbol{h_{k+1}}$ and output $\boldsymbol{y_{k+1}}$ through Eq.(25) and Eq.(12), respectively;

(d) Calculate $\boldsymbol{P}_{k+1}$ and $\boldsymbol{\beta}_{k+1}$ using Eq.(24) and Eq.(23), respectively, so as to get the new prediction model $\boldsymbol{f_{k+1}}$ dynamically;

(e) Set $k = k + 1$ and go to Step 2.

5. Optimization Algorithm-AWCPSO

5.1. Selection of the optimization algorithm

As mentioned above, the optimization algorithm also contributes significantly to the design optimization since it affects the reliability, usability, efficiency and computational cost of the entire optimization process. As mentioned above, gradient information and sensitivity analysis based traditional optimization algorithms are not suitable for complex optimization problem, e.g., when the number of optimization variables and the cost of function evaluation increase. Heuristic optimization algorithms such as GA, ANN and PSO do not need the continuity and derivative of objective function, which are better for global optimization problem. In the product design optimization problem, the optimization objective function and constraints obtained through the finite element analysis (FEA) are implicit as well as the relationship between input design variables and output performance parameters. It is generally difficult to judge whether the objective function is continuous or differentiable. Therefore, the heuristic optimization algorithm with greater stability and applicability is more suitable for the search of optimal design solution.

Based on the above analysis, the AWCPSO algorithm, which is improved on the basis of center particle swarm optimization (CPSO) [39] algorithm, is proposed to tackle the design optimization problem to further improve the efficiency and accuracy of the iterative optimization.

5.2. The AWCPSO algorithm

5.2.1. Theoretical basis

Suppose that there are *M* particles moving around in a *k*-dimensional search space, the *i*-th particle at the *t*-th iteration has a position $P_t(i) = X(i) = (x_1, x_2, \dots, x_k)$ corresponding to the *i*-th design scheme of *k* design variables, and P_{th} is the best position achieved so far. $V_t\ (i) = (v_1, v_2, \dots, v_k)$ represents current velocity, $P_w(i)$ is the weighted center. The velocity and position of the *i*-th particle at the (*t*+*1*)-th iteration will be computed based on the following equations:

$$V_t(i+1) = w_i V_t\ (i) + c_1 r_1\big(P_{th}(i) - P_t(i)\big) + c_2 r_2\big(P_w(i) - P_t(i)\big) \tag{26}$$

$$P_t(i+1) = P_t(i) + V_t(i+1) \tag{27}$$

Here, w_i is inertia weight, c_1 and c_2 $(c_1, c_2 > 0)$ are cognitive learning rate and social learning rate, respectively, and r_1 and r_2 are random number in the range [0,1]. In each dimension, the velocities of particles are confined within $[V_{imin}, V_{imax}]_{i=1}^{k}$. When any element of $V_t\ (i)$ exceeds the threshold V_{imin} or V_{imax}, they are set as the corresponding threshold. It means that $P_t(i)$ cannot exceed the search space (corresponding to the range of the design parameters).

The AWCPSO algorithm includes two strategies: (1) Weighted center $P_w(i)$ rather than simple arithmetic center $P_c(i)$ is adopted to reduce the risk of falling into local optimum, which is computed according to the contributions of different particles to extreme search; (2) Inertia weight w is adaptively adjusted based on the perception factor w_Δ to balance convergence accuracy and rate, which is set according to the premature convergence degree of population and individual fitness value to maintain the diversity of inertia weight.

5.2.2. Computation of the weighted center P_w

The arithmetic center P_c of all the particles is used as the global best solution P_g in CPSO algorithm. It is more stable and representative but ignores the differences between different particles. Actually, different particles have different contributions to extreme value search. In general, the particle close to global optimum with small fitness contributes more to the search process and thus should be paid more attention. Thus, the weighted center P_w is proposed to replace P_c to reflect the differences of individual particle. However, it is not appropriate to use particle fitness directly to compute P_w since the particle with small fitness value has greater impact and should have a larger weight.

Suppose that the maximum and minimum fitness value of the particle swarm are f_{max} and f_{min}. The fitness value and its normalized value of P_i in the *t*-th iteration are f_i and f_i^*, respectively, and φ_i represents its weight. Then $P_w(i)$ is computed as follows:

$$P_w(i) = \sum \varphi_i P_t(i)/M \tag{28}$$

$$\varphi_i = \frac{\sum_{i=1}^{M} f_i^* - f_i^*}{\sum_{i=1}^{M} f_i^*} \tag{29}$$

$$f_i^* = \frac{f_i - f_{min}}{f_{max} - f_{min}} \tag{30}$$

5.2.3. Computation of the inertia weight w

The inertia weight w describes the influence of previous particle velocity on current particle velocity. It is critical as it balances global exploration and local exploitation abilities of the swarm. Generally, w should be set as different values according to different situations to balance the search speed and accuracy.

Note that the maintenance of population diversity is a prerequisite to converge to global optimum under the predetermined convergence condition. In the PSO algorithm, the population diversity is gradually lost due to the continuous tracking of global optimal solution. Although it converges fast, it may lead to premature convergence. However, the convergence speed will be affected if the tracking of global optimal solution is reduced [40]. Thereby, the inertia weight perception factor w_Δ is adopted to evaluate the premature convergence degree of the particle swarm.

Let f_g^* represent the optimal fitness value in the *t*-th iteration, and the fitness value of the corresponding weighted center is f_w^*. Then, w_Δ is computed as:

$$w_\Delta = \frac{f_w^* - f_g^*}{f_w^*} \tag{31}$$

The smaller w_Δ indicates that the particle distribution is aggregated, and the particle swarm tends to converge prematurely. At this time, it is necessary to increase w_i to get a greater velocity to effectively escape from local optimum and avoid premature convergence. However, it is not suitable to apply the same adaptive operation to the whole particle swarm directly since the excellent particles may be destroyed with the increase of w_i.

Therefore, different adaptive operations are conducted based on their individual fitness values to maintain the diversity of the inertia weights. Here, w_i is computed as follows:

$$w_i = exp\left(\frac{f_i - f_{avg}}{f_{avg}}\right) \tag{32}$$

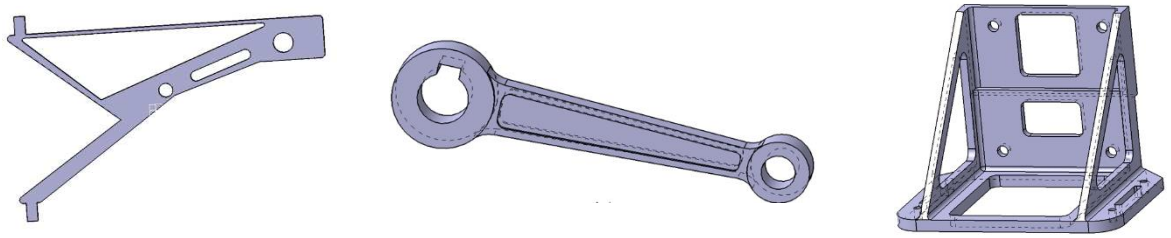
When the fitness value f_i of the i-th particle P_i is smaller than f_{avg}, P_i is a relatively good particle and thus should be assigned a smaller w_i to accelerate the convergence speed. When f_i is larger than f_{avg}, P_i should be given a larger weight to escape from local optimum to avoid premature convergence. In this way, different particles are assigned with different inertia weights to balance the search speed and accuracy comprehensively.

6. Experiments and Discussion

Three case studies are conducted to demonstrate the feasibility and effectiveness of the proposed approach.

6.1. Experimental data set preparation

Since industrial data set cannot be obtained easily, CATIA (CAD software) and Abaqus (CAE software) are applied to design and simulate models to construct artificial data set for the subsequent design optimization. Three parts from simple to complex including Automobile Component (AC), Torque Arm (TA), and Bracket Structure (BS), as shown in Fig. 3, are selected as the test objects. 4, 5 and 11 key variables which are critical for product performance are used to drive the parametric modeling process respectively, while other predefined geometrical parameters remain unchanged. The batch design of geometric models is based on the comprehensive or orthogonal experiments in the Statistical Product and Service Solutions (SPSS).



(a) Automobile component model (b) Torque arm model (c) Bracket structure model

Fig. 3 The CAD models of three test models

The comprehensive experimental design of 256 experiments (four factors with four levels) are conducted to design CAD models for the first part. But only 176 models are generated successfully for the subsequent simulation due to the sketch self-intersection caused by the conflicts between different parameters. The reason

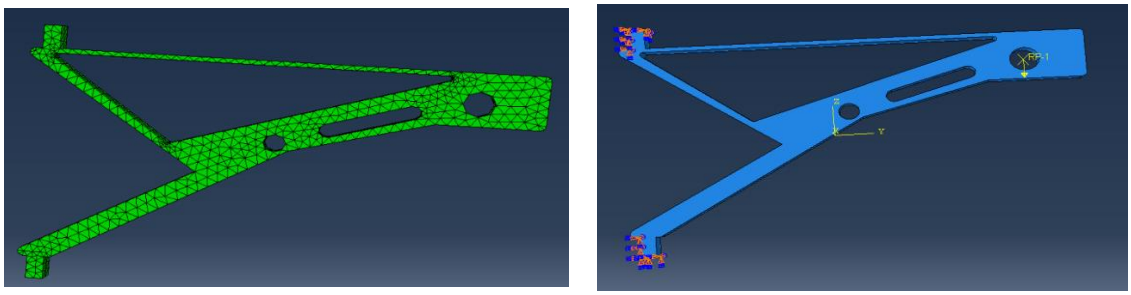
is that invalid parameter combinations cannot design a truly valid model, not to mention the subsequent analysis to obtain the simulation data. In the same way, the effective number for the last two parts are 243 and 145, respectively.

All the models are performed in the structural analysis with the aim to obtain more efficient and lighter structures. The entire process is conducted in the CAD/CAE integration framework. All the test parts were easily modeled, automatically updated and analyzed to obtain the training data set based on the changed input design variables. The related simulation experimental setting is listed in Table 1, and the corresponding tetrahedral mesh models and simulation models are shown in Fig. 4-Fig. 6. The original design and corresponding simulation data constitute the input data set for the dynamic data mining algorithm.

Table 1 The experimental setting of batch analysis of simulation models.

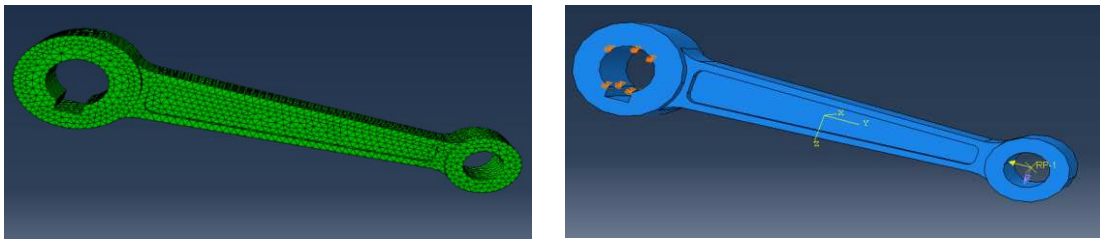
Part	Material properties	Interaction and load conditions
AC	E = 72GPa, v = 0.3	Fixed constraint at the upper and lower connection region on the left Concentration force F=5.4kN at the center of right end
TA	E = 200GPa, v = 0.3	Fixed constraint at the large end hole Bending moment P1=80kN, Compressive force P2=40kN at the center of small end
BS	E = 200GPa, v = 0.3	Fixed constraint at the four holes in the base Compressive force P1=47kN, Bending moment P2=42kN at the center of four screw holes

Note: E (Young's modulus) and v (Poisson's ratio).



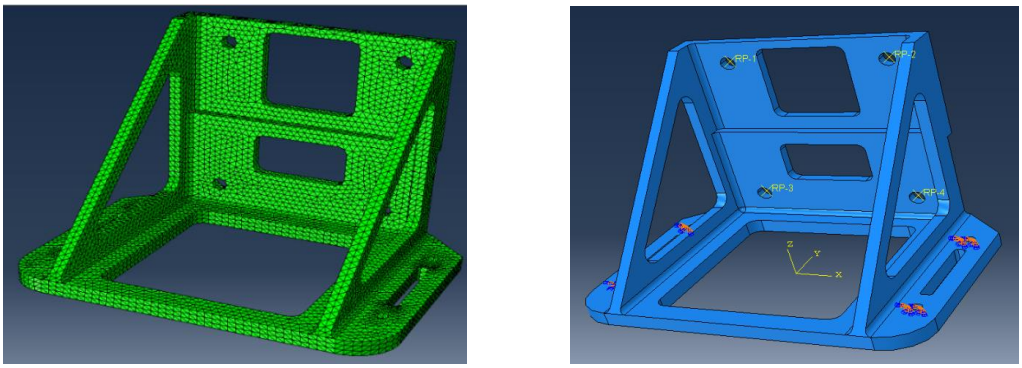
(a) Mesh model (b) Simulation model

Fig. 4 The mesh model and simulation model of the automobile component model.



(a) Mesh model (b) Simulation model

Fig. 5 The mesh model and simulation model of the torque arm model.



(a) Mesh model (b) Simulation model

Fig. 6 The mesh model and simulation model of the bracket structure model

6.2. Performance evaluation

6.2.1. Experimental setting

The optimization objective of test models is to minimize the model volume to reduce the material consumption while the maximum stress is constrained to be less than a given threshold to guarantee model strength. The related experimental parameters of prediction model and optimization model are set in Table 2.

Table 2 The experimental setting of prediction model and optimization model.

Part model	Number of hidden neurons	Activation function	MaxIt	Constraints
AC	Snum = 55, Vnum = 30	Sigmoid function	100	maxStress<=100MPa
TA	Snum = 35, Vnum = 30	Sigmoid function	100	maxStress<=120MPa
BS	Snum = 130, Vnum = 45	RBF function	200	maxStress<=320MPa

Where Snum and Vnum represent the number of hidden neurons for stress prediction model and volume prediction model, respectively. Two types of activation functions including Sigmoid function and RBF function are used for the subsequent model training and prediction. MaxIt, the maximum number of iterations, which is determined after initial trials, is set as the predetermined convergence condition. It is set differently

due to the complexity of optimization problem, i.e., MaxIt is set larger for the last case in order to better explore the optimal solution space of complex product. The maximum stress constraints are 100MPa, 120MPa and 320MPa, respectively.

It can be seen from the table, the numbers of hidden neurons of the first case are 55 and 30 for stress prediction model and volume prediction model respectively based on the experimental results of 10-folds cross-validation. Generally, the number of hidden neurons for the former is relatively larger since the design parameters and volume are more closely related and the design parameters can better describe the volume. In addition, the number of hidden neurons for the first two cases are relatively small, while it is larger for the last case to improve the prediction accuracy. Meanwhile, RBF function is selected as the activation function for the last case. This attributes to the fact that the number of design parameters are relatively large and the relationship between design parameters and performance parameters is relatively highly nonlinear.

6.2.2. Numerical analysis and comparison

In order to avoid the influence of initial particle swarm, a random sample set containing 1/4 of all the samples is selected for multiple independent optimization experiments under the predetermined convergence condition and constraints.

Firstly, 10 independent optimization experiments are conducted for the first model. The related numerical analysis results of optimal design solution are shown in Table 3. Where PV, TV, PS and TS represent the predicted and true volume and maximum stress, respectively. And REV and RES denote the relative error of volume and maximum stress. IterNum is the number of iterations required for optimization convergence.

Note that the optimization target value tends to be stable within specified number of iterations in most cases. Most of the experiments converge to the same or similar optimal design solution {30,370,180,6} after 20 to 40 iterations under the maximum stress constraint. The 2^{nd} and 5^{th} experiments are the optimal and worst optimization results, respectively. Although the iterative optimization processes are different, they eventually converge to the same optimal solution. Both the maximum stresses satisfy the predefined requirements while the volume difference is very small (only 0.581%=（363299-361190）/363299). Moreover, the relative error between the actual results and the predicted results of volume and stress are small. Especially for the volume prediction, i.e., the maximum relative error is almost less than 0.01%.

Table 3 The numerical analysis results of optimal design solution of 10 optimization experiments.

No.[D1,D2,D3,D4]	PV (cm^3)	TV(cm^3)	PS(MPa)	TS(MPa)	IterNum	REV(%)	RES(%)
1 [29.84,370,180,6.37]	362600	362581	99.8241	91.1449	60	0.0052	8.6945
2 [30,370,180,5.99]	361210	361190	100	99.3737	36	0.0055	0.6263
3 [30,170,180,6.28]	362280	362261	100	96.1195	27	0.0052	3.8805
4 [30,370,180,6.01]	361260	361245	100	94.7116	20	0.0042	5.2884
5 [30,370,180,6.56]	363340	363299	100	95.3227	21	0.0113	4.6773
6 [29.93,370,180,5.96]	361220	361202	99.4937	95.5895	45	0.0050	3.9241
7 [29.21,370,180,6.12]	361690	361670	99.8325	94.2519	25	0.0055	5.5900
8 [30,370,180,6.08]	361550	361513	99.841	95.3724	36	0.0102	4.4757
9 [30,370,180,6.03]	361360	361319	100	96.2159	35	0.0113	3.7841
10 [30,370,180,6.17]	362090	362067	100	94.959	40	0.0064	5.0410

Secondly, to further prove the superior performance of the proposed approach, comparative experiments are made between the proposed approach and the approach based on the original OS-ELM and PSO on the last two test models. The numerical analysis results of optimal design solution of comparative experiments are shown in Table 4. 10 independent optimization experiments are conducted for performance comparison to eliminate random factors and statistical influences. TA(a) and BS(a) represent the first five experiments based on the proposed approach, while TA(b) and BS(b) are the last five experiments based on the original OS-ELM and PSO.

Table 4 The numerical analysis results of optimal design solution of comparative experiments.

Part	No.	PV (cm^3)	TV (cm^3)	PS (MPa)	TS (MPa)	IterNum	REV(%)	RES(%)
TA(a)	1	446290	446421	105.1383	103.7121	30	-0.0294	1.3752
	2	445820	446421	103.1209	103.7100	25	-0.1347	-0.5700
	3	446760	446421	103.5012	103.7100	22	0.0760	-0.2034
	4	443590	446421	104.3256	103.7121	12	-0.6346	0.5915
	5	446730	446421	103.0982	103.7100	25	0.0693	-0.5919
TA(b)	1	445400	446421	105.3280	103.7121	40	-0.2289	1.5581
	2	446870	446421	102.4409	103.7100	40	0.1006	-1.2257
	3	447720	446421	105.0127	103.7100	40	0.2912	1.2540
	4	447760	446421	104.7487	103.7100	60	0.3001	0.9995
	5	448920	446421	102.8420	103.7121	90	0.5602	-0.8390
BS(a)	1	8.70E+06	8.65E+06	319.9907	311.8591	120	-0.00547	0.025412
	2	8.62E+06	8.64E+06	319.9900	307.4200	130	0.001686	0.039282
	3	8.59E+06	8.60E+06	319.9932	317.3222	140	0.001055	0.008347
	4	8.65E+06	8.66E+06	320.0000	294.5478	60	0.001088	0.079538
	5	8.61E+06	8.61E+06	320.0000	320.0880	60	-0.00021	-0.00028
BS(b)	1	8.68E+06	8.73E+06	320.0000	307.3220	200	-0.00519	0.041253
	2	8.75E+06	8.82E+06	319.9794	293.9997	220	-0.00723	0.088366
	3	8.86E+06	8.89E+06	320.0000	305.8600	260	-0.00285	0.04623
	4	8.64E+06	8.67E+06	320.0000	311.5201	250	-0.00370	0.027221
	5	8.95E+06	9.03E+06	320.0000	298.5201	280	-0.009	0.071955

The first five experiments for the TA model converge to optimal design solution {3,25,90,30,12} with less iterations (20~30) even with random initial particle swarm. However, the last five ones converge to optimal design solution more than 40 iterations in most cases. The experiments of the BS model converge to the same optimal design solution {25,120,200,40,220,160,20,25,160,350,300}. But the last five experiments

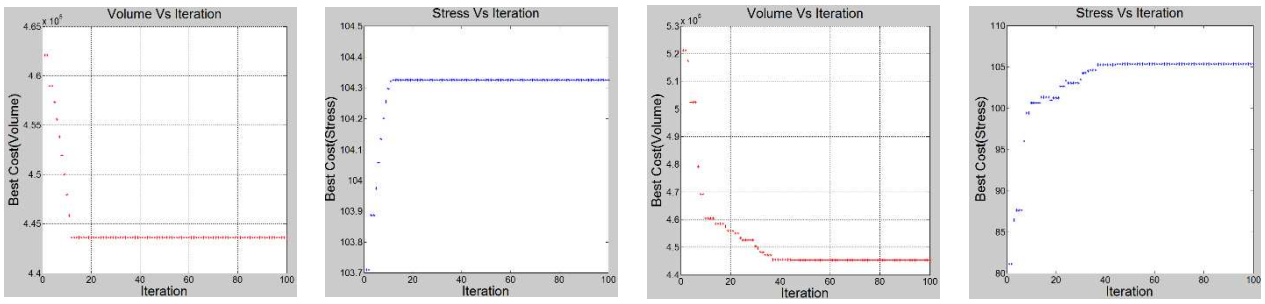
converge very slowly, wherein the number of iterations is basically more than 200. It can be inferred that the convergence efficiency is improved to some extent by using the AWCPSO algorithm since the optimization time increases with the number of iterations.

Similarly, the prediction accuracy is also improved by using $W_{adaptive}$OS-ELM algorithm. The relative error of volume of the proposed approach is most less than 0.2% for the TA model, while it is more than 0.2% for the original algorithm in most instances. The relative error of maximum stress is relatively large, but the proposed approach shows better prediction performance with lower prediction error. The numerical results of the BS model are consistent with the TA model. The prediction accuracy of volume and stress of the proposed approach are relatively higher, i.e., most of the relative errors of stress are less than 0.04%, while they are larger for the last five experiments.

In addition, the iterative convergence speed of the BS model is relatively slow in comparison with the previous two models since the design solution space with more design parameters is relatively large. Generally, the more iterations, the longer the optimization time. But the whole convergence process is essentially instantaneous due to the fast objective performance evaluation of dynamic prediction model and the use of improvement strategies of AWCPSO algorithm.

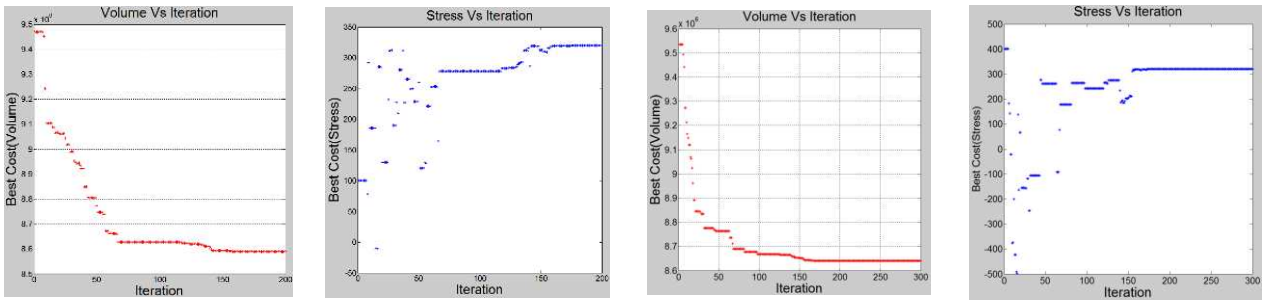
6.2.3. Visualization analysis and comparison

The corresponding iterative optimization processes of volume and stress of above comparative experiments are given in Fig. 7 and Fig. 8. Only the best experimental results are listed. All the experimental results converge to nearly the same optimal solution within the specified number of iterations regardless of different iterative process. But the proposed approach converges faster and the optimization results are better, which are consistent with the numerical analysis results in Table 4.



(a) The proposed approach in this study (b) Approach based on original OS-ELM and PSO

Fig. 7 The iterative optimization process of comparative experiments for the TA model.

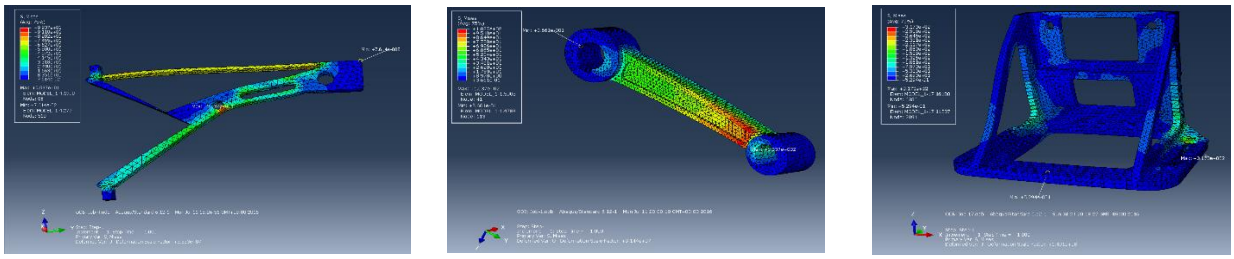


(a) The proposed approach in this study (b) Approach based on original OS-ELM and PSO

Fig. 8 The iterative optimization process of comparative experiments for the BS model.

The actual simulation results optimal design solution of all the test models are shown in Fig. 9. For the optimal design solution of the 1st test model, the actual volume and maximum stress are 361190cm^{3}and 99.37Pa, and the relative error are 0.0055% and 0.6263%, respectively, with respect to the corresponding predicted results. The prediction model is suitable for performance evaluation of new particle during the optimization process due to its high prediction accuracy. The volume and maximum stress of the optimal design solution for the BS model are 8599270cm^3 and 317.3222Mpa, respectively. And the relative errors are nearly 0.0011% and 0.0083%, it is unnecessary to further optimize the prediction model since the differences between the actual simulation results and the prediction results are small and the actual results satisfy the optimization goal. Only the simulation data set corresponding to the optimal design solution are taken as the new samples to dynamically update the prediction model for the next optimization evaluation.

Based on the above observation, the prediction model with high prediction accuracy is reliable and valuable as a reference value for performance evaluation in the optimization process. The proposed approach is feasible and effective in seeking the optimal design solution automatically and efficiently. Especially, the application of dynamic prediction model and optimization algorithm greatly reduces the number of actual simulations and improves the iterative optimization efficiency with less iterations.



(a) Automobile component model (b) Torque arm model (c) Bracket structure model

Fig. 9 The actual simulation results of optimal design solution of test models.

7. Conclusion and Future Work

In this study, a simulation data driven design approach is proposed to support the rapid product optimization effectively. The main work is summarized as follows:

(1) A simulation data driven design approach which combines dynamic simulation data mining and design optimization is proposed to search the optimal design solution. In this way, the optimal design solution can be searched automatically with less actual simulation iterations and higher iterative optimization efficiency.

(2) An incremental learning based dynamic data mining algorithm—$\mathrm{W}_{\mathrm{adaptive}}$ OS-ELM algorithm is proposed to construct the dynamic prediction model to evaluate the performance of new design scheme quickly and accurately. The prediction model is updated incrementally through new optimum data set to effectively simulate the dynamics and complexities of the real simulation system, and thus to reduce the modelling complexity, improve the prediction accuracy and efficiency.

(3) An improved optimization algorithm—AWCPSO algorithm is proposed to guide the search for good design solution automatically to reduce the professional experience requirements. The weighted center $P_w(i)$ and inertia weight w are adopted to accelerate the entire product optimization process by improving convergence accuracy and convergence speed.

Extensive experimental results demonstrate the feasibility, effectiveness and correctness of the proposed approach for product optimization. But only incorporating "good" data sets may also restrict potential design solutions by biasing the training data. In the future, random sampling method will be considered to randomly select a part of potential design schemes to increase the diversity and randomness of the incremental samples. Meanwhile, the dynamic data mining and optimization module are replaceable, a more extensive knowledge base algorithm can be built to solve the practical optimization problems of different characteristics. Large-scale industrial data set for more complex parts and assemblies with different optimization problems are needed to further demonstrate the performance of the approach.

Acknowledgement

The authors appreciate the support by the National Key technology R&D Program (No. 2018YFB1700901), and the National Science Foundation of China (No. 61702517, 61772525, 61873236, 61802211, 61902345).

References

[1] Roy R, Hinduja S, Teti R. Recent advances in engineering design optimisation: Challenges and future trends[J]. CIRP Annals - Manufacturing Technology, 2008, 57(2):697-715.

[2] Park H S, Dang X P. Structural optimization based on CAD-CAE integration and metamodeling techniques[J]. Computer-Aided Design, 2010, 42(10):889-902.

[3] Wang D , Hu F , Ma Z , Wu Z & Zhang W. A CAD/CAE integrated framework for structural design optimization using sequential approximation optimization[J]. Advances in Engineering Software, 2014, 76:56-68.

[4] Cho C S , Choi E H , Cho J R , & Lim O K. Topology and parameter optimization of a foaming jig reinforcement structure by the response surface method[J]. Computer Aided Design, 2011, 43(12):1707-1716.

[5] Islam M, Buijk A, Rais-Rohani M, & Motoyama K. Process parameter optimization of lap joint fillet weld based on FEM–RSM–GA integration technique[J]. Advances in Engineering Software, 2015, 79(C):127-136.

[6] Kang G J, Park C H, Choi D H. Metamodel-based design optimization of injection molding process variables and gates of an automotive glove box for enhancing its quality[J]. Journal of Mechanical Science and Technology, 2016, 30(4):1723-1732.

[7] Lechevalier D, Hudak S, Ak R, Tina Lee Y, & Foufou S. A neural network meta-model and its application for manufacturing[C]// IEEE Big Data Conference. IEEE, 2015:1428-1435.

[8] Nguyen A T, Reiter S, Rigo P. A review on simulation-based optimization methods applied to building performance analysis[J]. Applied Energy, 2014, 113(6):1043-1058.

[9] Dai L, Guan Z Q, Chen B S, & Zhang H W. An open platform of shape design optimization for shell structure[J]. Structural & Multidisciplinary Optimization, 2008, 35(6):609-622.

[10] Hare W, Nutini J, Tesfamariam S. A survey of non-gradient optimization methods in structural engineering[J]. Advances in Engineering Software, 2013, 59(5):19-28.

[11] Corriveau G, Guilbault R, Tahan A. Genetic algorithms and finite element coupling for mechanical optimization.[J]. Advances in Engineering Software, 2010, 41(3):422-426.

[12] Li W, Mcadams D A. Designing Optimal Origami Structures by Computational Evolutionary

Embryogeny[J]. Journal of Computing & Information Science in Engineering, 2014, 15(1):V05BT08A035-V05BT08A035.

[13] Kou X Y, Parks G T, Tan S T. Optimal design of functionally graded materials using a procedural model and particle swarm optimization[J]. Computer-Aided Design, 2012, 44(4):300-310.

[14] Flocker F W, Bravo R H. Ensuring Global Convergence in Design Optimization using the Particle Swarm Optimization Technique[J]. Journal of Mechanical Design, 2016, 138(8).

[15] Hu M , Wu T , Weir J D . An Adaptive Particle Swarm Optimization With Multiple Adaptive Methods[J]. IEEE Transactions on Evolutionary Computation, 2013, 17(5):705-720.

[16] Yıldız A R. An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry[J]. Journal of Materials Processing Technology, 2009, 209(6):2773-2780.

[17] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007, 39(3):459-471.

[18] Wang D, Wu Z, Fei Y, & Zhang W. Review: Structural design employing a sequential approximation optimization approach[J]. Computers & Structures, 2014, 134(4):75-87.

[19] Wang G G, Shan S. Review of Metamodeling Techniques in Support of Engineering Design Optimization[J]. Journal of Mechanical Design, 2007, 129(4).

[20] Chen T Y, Huang J H. Application of data mining in a global optimization algorithm[J]. Advances in Engineering Software, 2013, 66(12):24-33.

[21] Better M, Glover F, Laguna M. Advances in analytics: Integrating dynamic data mining with simulation optimization[J]. Ibm Journal of Research & Development, 2007, 51(3.4):477-487..

[22] Li Y, Roy U. Challenges in Developing a Computational Platform to Integrate Data Analytics with Simulation-Based Optimization[C]// ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. 2015.

[23] Li X, Shao Y, Liu Y. Takagi-Sugeno Model Based Simulation Data Mining for Efficient Product Design[C]// ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. 2015:V01BT02A032.

[24] Zhang H R. Incremental and Online Learning Algorithm for Regression Least Squares Support Vector

Machine[J]. Chinese Journal of Computers, 2006, 29(3):400-406.

[25] Gu B, Sheng Victor S, Wang Z, Ho D, Osman S, &Li S. Incremental learning for ν -Support Vector Regression[J]. Neural Networks[J], 2015, 67:140-150.

[26] Gjerkes H, Malensek J, Sitar A, & Golobic I. Product identification in industrial batch fermentation using a variable forgetting factor[J]. Control Engineering Practice, 2011, 19(10):1208-1215.

[27] Liang N Y, Huang G B, Saratchandran P, & Sundararajan N. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks[J]. IEEE Transactions on Neural Networks, 2006, 17(6):1411-23.

[28] Soares S G, Rui A. An on-line weighted ensemble of regressor models to handle concept drifts[J]. Engineering Applications of Artificial Intelligence, 2015, 37:392-406.

[29] S.G. Soares, R. Araújo, A dynamic and on-line ensemble regression for changing environments, Expert Syst. Appl. 42 (6) (2015) 2935–2948.

[30] Brzezinski D, Stefanowski J. Combining block-based and online methods in learning ensembles from concept drifting data streams[J]. Information Sciences An International Journal, 2014, 265(5):50-67.

[31] G-B．Huang，L．Chen，and C．-K．Siew．Universal approximaion using incremental constructive feedforward networks with random hidden nodes．IEEE Transaction on Neural Network，2006，1 7(4)：879-892.

[32] Rong H J, Huang G B, Sundararajan N, & Saratchandran P. Online Sequential Fuzzy Extreme Learning Machine for Function Approximation and Classification Problems[J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society, 2009, 39(4):1067-1072.

[33] Huynh H T, Won Y. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks[J]. Pattern Recognition Letters, 2011, 32(14):1930-1935.

[34] Scardapane S, Comminiello D, Scarpiniti M, & Uncini A. Online Sequential Extreme Learning Machine With Kernels[J]. IEEE Trans Neural Netw Learn Syst, 2015, 26(9):2214-2220.

[35] Lan Y, Soh Y C, Huang G B. Letters: Ensemble of online sequential extreme learning machine[J]. Neurocomputing, 2009, 72(13–15):3391-3395.

[36] K. S. Banerjee. Generalized Inverse of Matrices and Its Applications[M]. Wiley, 1973.

[37] He X A, Liu W. Preliminary Discussion on Weighted Least Square Method and Its Residual Plot: Also Answering Associate Professor Sun Xiaosu[J]. Statistical Research, 2006.

[38] Higham B N J. Accuracy and Stability of Numerical Algorithms (Second Edition[J]. 2002.

[39] Liu Y, Qin Z, Shi Z, & Lu J. Center particle swarm optimization[J]. Neurocomputing, 2007, 70(4–6):672-679

[40] Han J H, Zheng-Rong L I, Wei Z C. Adaptive Particle Swarm Optimization Algorithm and Simulation[J]. Journal of System Simulation, 2006, 18(10):2969-2971.

Figure list

Table list